Vysoká škola báňská – Technická univerzita Ostrava

Fakulta elektrotechniky a informatiky

Katedra informatiky

Telemetrie pro dálkově řízené modely
Telemetry System for RC Models

2015 **Michal Matula**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Michal Matula**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Telemetrie pro dálkově řízené modely
Telemetry System for RC Models

Zásady pro vypracování:

Cílem bakalářské práce je navrhnout a realizovat hardwarový modul pro vzdálený přenos telemetrických dat. Přenos dat bude probíhat v ISM pásmu pomocí některého z dostupných rádiových modulů. Připojení k počítači bude realizováno pomocí přijímače připojeného na USB nebo sériový port. Pro modul vysílače bude navržen vhodný komunikační protokol a ten bude implementován v dostupném jednočipovém počítači (Atmel, Microchip apod.)

1. Rešerše telemetrických systémů a ISM modulů, jejich vzájemné srovnání.
2. Návrh komunikačního protokolu pro odesílání dat, zabezpečení přenosu proti chybám.
3. Implementace komunikačních rutin do vybraného jednočipového počítače.
4. Implementace přijímacího a dekódovacího software pro stolní počítač.
5. Otestování navrženého řešení (spolehlivost a dosah).

Seznam doporučené odborné literatury:

[1] Paul Scherz, Practical Electronics for Inventors, Tab Electronics; 3 edition, 2013, ISBN 978-0071771337
[2] Frank Carden, Telemetry Systems Engineering, Artech House; 2nd edition edition, 2002, ISBN 978-1580532570
[3] Vladimír Váňa, Atmel AVR: Programování v jazyce C, BEN-Technická literatura, 2003, ISBN 80-7300-102-0

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Mgr. Ing. Michal Krumnikl, Ph.D.**

Datum zadání: 01.09.2014
Datum odevzdání: 07.05.2015

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto bakalářskou práci na téma telemetrie pro dálkově řízené modely vypracoval samostatně pod vedením vedoucího bakalářské práce Mgr. Ing. Michala Krumnikla, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne: *4. 5. 2015*

.....
podpis studenta

Poděkování

Rád bych poděkoval Mgr. Ing. Michalu Krumniklovi, Ph.D. za odbornou pomoc, obrovskou vstřícnost a cenné rady při vytváření této bakalářské práce. Také bych chtěl poděkovat Bc. Luboši Matějíčkovi za věcné připomínky a pomoc při krizových momentech bakalářské práce a přátelům a rodině za podporu, která přispěla ke zdárnému dokončení práce.

Abstrakt

Tato bakalářská práce je zaměřena na návrh komunikace hardwarového modulu v předem stanoveném bezlicenčním pásmu. Hlavní pozornost je věnována navržení obecného komunikačního protokolu a sestavení modulu pro měření telemetrických veličin. K tomu je použitá elektronická součástka akcelerometr a gyroskop. Data jsou následně přeposílána na sériový port. Přijímání a interpretace veličin bude zpracována ve vytvořeném softwaru pro stolní počítač. Dále se tato práce zaměřuje na srovnání jednotlivých komunikačních technologií a otestování funkčnosti výsledného návrhu v praxi.

Klíčová slova

telemetrie; akcelerometr; gyroskop; sériový port; komunikační protokol; bezlicenční pásmo; SPI; I2C; COM

Abstract

This bachelor thesis focuses on the design of hardware module communication in pre-established licenceless zone. Main attention is dedicated to the design of general communication protocol and construction of a module for telemetric quantities measurement. Electronic parts called accelerometer and gyroscope are used for the measuring. Dates are subsequently sent to serial port. Acquisition and interpretation of measured quantities will take place in software created for personal computer. Furthermore, the thesis deals with comparison of particular communication technologies and verification of the final design function in practice.

Key words

telemetry; accelerometer; gyroscope; serial port; communication protocol; unlicensed zone; SPI; I2C; COM

Seznam použitých zkratek

WPF	Windows Presentation Foundation
SPI	Serial Peripheral Interface
I2C	Inter Integrated Circuit
ISM	Industrial, scientific and medical
C#	C Sharp
RC	Radio Control
CRC	Cyclic redundancy check
COM	Communication port
TRU	Transmitter Receiver Unit
GFSK	Gaussian frequency-shift keying
EEPROM	Electrically Erasable Programmable Read-Only Memory
PWM	Pulse Width Modulation
UART	Universal Synchronous Asynchronous Receiver and Transmitter

Obsah

Seznam tabulek

Seznam obrázků

Úvod

Telemetrie je technologie používána k měření a dálkovému přenosu dat. Dálkový přenos dat bývá nejčastěji realizován pomocí rádiového nebo infračerveného signálu. V dnešní době hrají telemetrické systémy obrovskou roli především v kosmonautice a meteorologii. Toto téma bakalářské práce jsem si zvolil, protože ve mně už dávno vzbudila střední škola zájem o tyto systémy a její propojení s informační technologií. Práce mi pomůže lépe porozumět problematice bezdrátové komunikace, programování jednočipových mikropočítačů a snímání telemetrických údajů. Mým cílem bude navrhnout modul pro snímání telemetrických veličin, komunikující pomocí obecného protokolu. Modul by měl být použitelný pro více druhů elektronických snímačů.

Začátek práce je věnován seznámení se s bezlicenčním ISM pásmem pro bezdrátovou komunikaci. Poté jsou porovnána jednotlivá ISM pásma spolu s technologiemi pro vytvoření bezdrátové sítě. Na základě těchto srovnání budou vybrány součástky pro bezdrátovou komunikaci a snímání telemetrických veličin. Pozornost je věnovaná také zapojení těchto součástek.

Dále bude popsán způsob komunikace mezi senzory, mikrokontroléry a RF moduly pro odesílání a příjmem dat. Pomocí senzorů jsou snímány telemetrické údaje, které jsou mikrokontrolérem předány vysílacímu modulu. Vysílací modul má za úkol prostřednictvím navrženého komunikačního protokolu odeslat telemetrická data přijímacímu modulu. Při návrhu komunikačního protokolu je kladen důraz na jeho obecný formát, aby bylo možné obsluhovat co největší množství senzorů. Přijímací část modulu je připojena k sériovému portu počítače.

Přes tento port se budou přijatá data zpracovávat a vyhodnocovat v navržené aplikaci. Tato aplikace by měla být co nejvíce přehledná, aby měl uživatel přehled o tom, jaké hodnoty jsou snímány a následně zobrazovány.

Navržený modul bude testován v různém prostředí a bude zkoumána nejen jeho spolehlivost, ale také schopnost modulu komunikovat na větší vzdálenosti.

1. Porovnání telemetrických systémů

V dnešní době existuje celá řada technologií a zařízení pracujících na principu snímání telemetrických dat. Telemetrické systémy jsou rozšířeny do všech možných oblastí, např. letecký a automobilový průmysl, modelářství, kosmonautika nebo zemědělství. V bakalářské práci se věnuji telemetrickým systémům pro RC modely. Na trhu existuje poměrně velká část systému a vybrat některé konkrétní není jednoduché. Jedna z možností jak rozlišit jednotlivé systémy spočívá v tom, jak jsou závislé na RC soupravě. Existují systémy zcela nezávislé, polo závislé nebo zcela integrované systémy do RC souprav.

Jako příklad nezávislého systému můžu uvést Quanum telemetry[1]. Jde o samostatnou jednotku napájenou z pohonného akumulátoru a měřící veškeré jeho veličiny. Jedná se o jednoduchý systém s malým dosahem, kdy data jsou zobrazována na malé obrazovce.

Mezi představitele polo nezávislého systému můžu uvést telemetrii FrSky[2]. Rozdíl spočívá v tom, že telemetrie obsahuje vysílací a přijímací modul starající se o přenos dat. Sběrem dat je pověřen senzor hub, na který jsou připojeny jednotlivé senzory. Hub je poté propojen s přijímačem. Data jsou zobrazována na display a ten je připojen kabelem k vysílacímu modulu.

Třetí integrované systémy jsou nejjednodušší. Jejich používání je uživateli maximálně zjednodušeno. Představitelem je např. systém Spektrum[3]. Základem systému je telemetrická jednotka spolu se základními senzory a kabely pro jednotlivé propojení. Telemetrická jednotka se propojí s přijímačem a celý systém se spáruje s rádiem. Tento systém také nabízí interface, kdy se dají data zobrazit nejen na samotném rádiu, ale také na iPhonu nebo iPadu. Náš koncept telemetrického modulu vychází z polo nezávislého systému, kdy telemetrické senzory bude mít na starosti mikrokontrolér a k němu připojen vysílač. Ten odešle paket s daty přijímači napojeném na mikrokontrolér číslo dvě, který je připojen na zobrazovací zařízení v mém případe notebook. Rozdíly mezi systémem FrSky a mým vytvořeným modulem jsou jak v jejich rozsahu použití, tak především v jejich ceně. Mým úkolem bude vypracovat telemetrický modul podobného konceptu jako např. telemetrie FrSky s tím rozdílem, že cena této telemetrie se pohybuje v řádech tisíců korun a můj telemetrický modul bude levnější, a to v řádech stovek korun.

Při vybrání technologií pro přenos dat jsem se rozhodoval mezi technologiemi pro komunikaci ZigBee nebo RF24Network. Technologie RF24Network vychází z technologie ZigBee avšak není její přímou implementací. Rozdíly mezi technologiemi jsou následující: ZigBee je robustnější technologie s bohatou sadou protokolů a podporou pro velké množství čipů. V závislosti na druhu čipu probíhá komunikace ve třech bezlicenčních pásmech a to 868

1http://www.quanum-rc.com/product-telesys.php

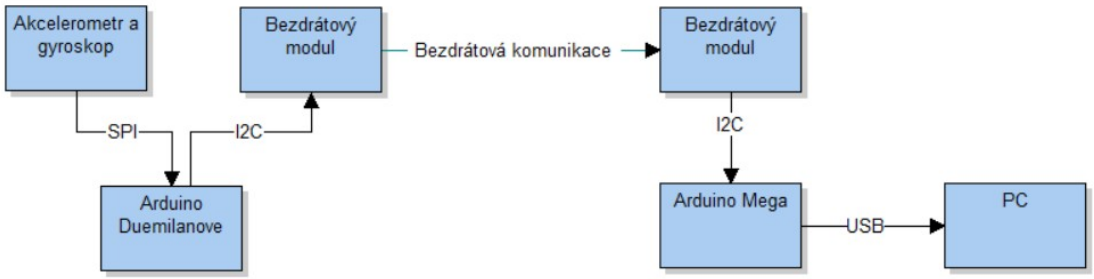
2http://www.frsky-rc.com/

3http://www.spektrumrc.com/

až 868,6 MHz pro Evropu, 902 až 928 MHz pro Severní Ameriku, Austrálii a 2400 až 2483,5 Mhz celosvětově. Čipy pro technologii RF24Network jsou schopny komunikovat pouze v celosvětovém pásmu 2400 až 2483,5 Mhz. RF24Network je však velmi levná technologie a ceny čipů se pohybují okolo 2$ oproti 20 $ za čipy podporující technologii ZigBee. Obě technologie podporují stromovou topologii sítě, kde koordinátor, v našem případě mikrokontrolér, spouští komunikaci a udává parametry sítě, ale také topologii typu hvězda, kde uzly komunikují přímo s koordinátorem. ZigBee navíc podporuje také topologii typu síť, ve které je umožněno komunikovat mezi uzly na stejné úrovni. Obě technologie podporuji přepnutí koncových uzlů do režimu spánku. ZigBee uzly jsou nakonfigurovány pomocí AT příkazů nebo samotnou aplikaci pro Windows. U technologie RF24Network jsou uzly nakonfigurovány překompilováním firmwaru nebo zápisem do paměti EEPROM. Obě technologie používají pro odhalení chyb vzniklých během přenosu cyklických kódů CRC. U obou technologií jde o šestnáctibitový polynom ve tvaru 0xFFFF. Po zvážení všech pro a proti se spíše přikláním k technologii RF24Network. Tato technologie podporuje téměř veškeré funkce, potřebné pro svou bakalářskou práci. Navíc mě láká vyzkoušet si práci s alternativní technologií, která není tak světově rozšířená jako ZigBee. A v neposlední řadě je využíváni technologie RF24Network levnější. V kapitole 2.2 bude přihlíženo při výběru RF vysílače a přijímače na toto srovnání.

2. Výběr a sestavení modulu

Celý modul pro snímání veličin, zpracovávání, posílání a zobrazování hodnot se skládá z několika funkčních celků, senzorů pro snímání telemetrických veličin, mikroprocesoru, bezdrátových modulů a stolního počítače. Princip zařízení je založen na snímání hodnot pomocí senzoru, který bude na vyžádání posílat data do mikroprocesoru. Ten tyto údaje převede do formátu vhodného pro bezdrátový přenos a odešle data pomocí námi zvoleného vysílače vzduchem k přijímači. Přijímač je napojen na druhý mikroprocesor, který přijatá data zpracuje a pomocí kabelu přesune na stolní počítač. V počítači jsou údaje dále zpracovány vytvořenou aplikací a ta telemetrické údaje zobrazí. Jednotlivá komunikace mezi zařízeními probíhá pomocí SPI a I2C rozhraní, ty jsou popsány v kapitole 2.8. Přesun dat mezi mikroprocesorem a počítačem je realizován pomocí USB kabelu a následným přenesením na sériový port COM.



Obrázek 2.1: Blokové schéma zařízení

2.1 ISM pásmo

ISM (industrial, scientific and medical) jsou frekvenční pásma určená pro vysílání radiového signálu v oborech průmyslu, vědy a zdravotnictví. Vznikla za účelem odstranění nelegálního vzniku vysílačů. Byla vydána určitá pravidla, která měla zamezit tomuto vzniku a provozu nelegálních vysílačů. O tato pravidla, jejich dodržování a spravováni se stará ČTÚ (Český Telekomunikační Úřad). ČTÚ se zodpovídá ITU (Mezinárodní telekomunikační unií). Správa pásem musí být v souladu se závazky vyplývajícími z mezinárodních smluv. Frekvenční pásmo je rozděleno do dvou druhů: licenční a bezlicenční. Aby mohl provozovatel vysílat v licenčním pásmu, musí mít patřičnou licenci. Tyto pásma a licence jsou zpoplatněny velmi vysokou částkou. Stát poté chrání provozovatele, aby nebylo možné vysílat na totožné frekvenci jinou osobou. Vzhledem k rozšíření přístrojů pro běžné využívání, vzniklo pásmo sdružující frekvence pro volné použití neboli bezlicenční. Při provozu v tomto pásmu je však nutné dodržovat podmínky určené ČTÚ. Bezlicenční pásmo je definováno ve frekvenčním rozsahu od nejnižšího pásma 9-14 Khz až po nejvyšší 59,3-62 GHz. Pro účely mé práce se zaměřím na frekvence: 433 Mhz, 868 Mhz a také 2,4 Ghz. Tyto frekvence se liší maximálním vyzářeným výkonem, jak je uvedeno v tabulce.

Frekvence	Maximální vyzářený výkon
433 MHz	10mW
868 Mhz	25mW
2,4 GHz	100mW

Tabulka 2.1: Vybrané frekvence a jejich výkon

Pásmo 433 Mhz je určeno pro přenos malých objemů dat. Je využíváno především u levných bezdrátový myší, klávesnic nebo domácích meteostanic. Maximální vyzařovací výkon je podle všeobecného oprávnění využívání rádiových kmitočtů a k provozování zařízení krátkého dosahu 10 mW. S tím souvisí také maximální vzdálenost, která je teoreticky možná pro komunikaci a to je okolo 300 m. Problémem však je velké využití tohoto pásma a z toho důvodů vzniká množství rušivých signálů.

Pásmo 868 Mhz má oproti pásmu 433 několik výhod. Hlavní výhodou je možnost vysílat s větším maximálním vyzářeným výkonem a to 25 mW. Díky tomu lze vysílat do větší vzdálenosti, řádově o stovky metrů. Další nespornou výhodou je menší množství zařízení, které jsou provozovány na této frekvenci. Mezi nevýhody patří horší přenos signálu přes překážky a také nepatrně větší velikost modulu.

Pásmo 2,4 Ghz spadá pod standart IEEE 802.11. Tento standart dělí pásmo do jednotlivých kanálů obdobně, jako jsou rozdělena pásma pro televizní a rozhlasové vysílání. Pásmo 2,4000-2,4835 Ghz je rozděleno na 13 kanálů a ty jsou od sebe posunuty o 5 Mhz. Maximální vyzářený výkon dosahuje hodnoty 100 mW, čímž je možno vysílat až do vzdálenosti 1,6 km. Nevýhodou je velmi velké množství přístrojů komunikujících na stejné frekvenci a tudíž dochází k velkému rušení a zkreslení signálu.

2.2 RF vysílače a přijímače

Jedním z úkolů bylo také vybrat vhodný vysílač a přijímač pro komunikaci v bezlicenčním pásmu. Hlavními kritérii pro výběr byla cena, frekvenční pásmo pro komunikaci, výkon, spotřeba a komunikační rozhraní. Vzhledem k možnosti komunikace oběma směry, tedy od vysílače k přijímači a naopak, bylo upřednostňováno universální zařízení, s nímž je možno takto komunikovat. Toto zařízení nese název Transmitter Receiver Unit (TRU). Zařízení je ve své podstatě jak vysílač, tak přijímač dohromady. Existuje však způsob komunikace pouze jedním směrem a to od vysílače neboli Transmitteru směrem k přijímači neboli Recieveru. Spojením těchto dvou slov vznikl zkrácený název Transceiver. Výběr modulu pro komunikaci byl prováděn z několika druhů zařízení, uvedených v tabulce níže.

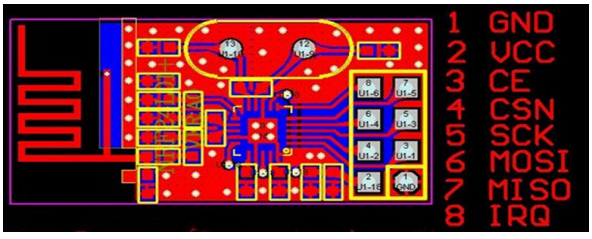
Parametr	Typ zažízení			
	MX-FS-03V	CC1101	Xbee-Pro	nRF24L01
Vcc (V)	2.8-3.6	1.8-3.6	2.8-3.4	1.9-3.6
RF výkon (dBm)	10	10	10	0
Frekvence (Mhz)	433	433/868/915	2400	2400
Přenovová rychlost (kbps)	32	500	250	2000
Citlivost přijímače (dBm)	-105	-110	-100	-82
Komunikačni rozhrani	Výrobce neuvádí	SPI	UART	SPI
Velikost (mm)	30x14	35x19	24x27	34x17
Cena (kč)	30	160	900	50

Tabulka 2.2: Tabulka srovnání vysílačů a přijímačů

Pro účely této bakalářské práce jsem se rozhodoval mezi Xbee-Pro a nRF24L01. Nakonec jsem vybral transceiver od firmy Nordic Semiconductor tedy model nRF24L01 [1]. Velkou výhodou tohoto TRU je velmi nízká spotřeba, rychlost přenosu dat, komunikační rozhraní SPI, velmi dobrá cena a také dostupnost. Při výběru jsem bral také ohled na srovnání technologií v kapitole 1. Bohužel jako jedna z nevýhod se zdá být menší vysílací výkon 1mW avšak vzhledem k zaměření této bakalářské práce není tento parametr klíčový.

2.3 Transceiver nRF24L01

Jednou z velkých výhod a důvod, proč jsem se rozhodl pro tento transceiver, je jeho velmi nízká spotřeba v šetřícím módu pouze 1 µA a klasickém 22 µA. Tento transceiver pracuje s napájecím napětím v rozmezí 1,9-3,6 V. Předpokládaný modul s mikrokontrolérem poskytuje napětí 3,3 V a 5 V. Tudíž mi napájecí napětí transceiveru plně vyhovuje a je také výhodou, že transceiver obsahuje rovněž vestavěný stabilizátor napětí. Také můžu nastavit provozní rychlost na 250 Kbps, 1 a 2 Mbps. Je zde také využívána modulace GFSK a zařízení vysílá na frekvenci 2,4 Ghz přes vestavěnou anténu. Programovatelný výstupní výkon je 0, -6, -12 nebo -18 dBm. Transceiver může vysílat pakety s dynamickou velikostí datové části a to od 1 až po 32 B. Také obsahuje hardwarově řešenou kontrolu paketů CRC, která může být nastavena jako osmibitová nebo šestnáctibitová. Aby bylo možné toto zařízení připojit k modulu s mikrokontrolérem je zde použit standardizovaný Dual in-line package soket (DIP). Přes tento soket probíhá komunikace s mikrokontrolérem přes rozhraní SPI, které bude popsáno v kapitole 2.8.1. Na obrázku je možno vidět rozdělení jednotlivých pinů pro transceiver nRF24L01.



Obrázek 2.2: Rozložení pinů pro transceiver

Transciever obsahuje celkově 8 pinů. Pin č.1 slouží pro připojení uzemnění. Pin č.2 slouží pro připojení napájecího napětí. Modul je schopen pracovat s napájecím napětím jak 5 tak 3,3 V avšak napětí větší než zmíněných 3,3 se nedoporučuje. Pin č.3 je určen pro výběr módu, zda bude transceiver vysílat nebo přijímat data. Piny č.4 – č.7 slouží pro SPI sběrnici, jejich zapojení se může měnit v závislosti na druhu mikrokontroléru, na který je transceiver připojen. Pin č.8 slouží pro přerušení a zapojuje se pouze pokud je přerušení vyžadováno, jinak není povinno tento pin zapojovat.

2.4 Telemetrické senzory

Telemetrické senzory se využívají v celé řadě odvětví, např. letectví, zdravotnictví, vojenství, ale také v modelařství. A právě tato odvětví slouží jako největší zdroj senzorů a informací vhodných pro naší práci. Existuje spousta druhů senzorů jako např. senzory teploty, otáček, výšky a varia, napětí, přetížení a náklonu nebo také GPS. Mě nejvíce zaujal druh senzorů, který se zaměřuje na měření náklonu a přetížení, neboli akcelerometr.

2.4.1 Popis akcelerometru

Akcelerometr [2] je zařízení, které se používá nejen k měření hodnoty statického nebo dynamického zrychlení, ale také se dají použít pro měření odstředivých a setrvačných sil a určování pozice tělesa díky jeho naklonění nebo vibracím. Velké uplatnění je především v leteckém průmyslu pro měření přetížení, jaké působí na pilota a také letadlo. Dále je používán v automobilovém průmyslu pro analýzu problémů v motoru použitím vibračních zkoušek, ale své uplatnění má také v elektrotechnice, kde slouží v počítačích jako ochrana harddisků před poškozením, například pádem notebooku. Existuje spousta druhů akcelerometrů, některé využívají piezoelektrického jevu a to takovým způsobem, že mikroskopické krystalové struktury obsažené uvnitř akcelerometru jsou namáhány způsobeným zrychlením a tím začnou generovat napětí. Další způsob je založen na snímání kapacitně citlivých buněk, neboli kapacitance. Kapacitně citlivým buňkám říkáme g-cell buňky. Tato buňka je založena na struktuře polovodičových materiálů. Při pohybu dochází k vychýlení pohyblivé části z její klidové polohy a tím dochází ke změně kapacitance. K tomu se přidají obvody pro úpravu signálů a ty mají za úkol přeměnit kapacitanci na napětí. Akcelerometry se dělí také na analogové a digitální což znamená, v jaké podobě bude mít akcelerometr výstupní data. U analogových akcelerometrů je spojité napětí úměrné velikosti zrychlení. Například pro hodnotu přetížená 0 g muže akcelerometr udávat hodnotu napětí 2,7 V pro 0,2 g je to 2,8 V a pro 0,4 g například 2,9 V. Analogové akcelerometry jsou proto využívány zejména pokud máme zařízení disponující analogovými vstupy. U digitálních akcelerometrů se využívá obvykle PWM neboli pulzní šířkové modulace. To znamená, že vlny o jisté frekvenci a množství času na maximální napěťové úrovni bude úměrné velikosti zrychlení. Tyto akcelerometry jsou proto využívány pokud pracují ve spojení s mikrokontrolérem s digitálními vstupy. Vzhledem k druhu použitého mikrokontroléru, o kterém se dozvíte v kapitole 2.6, jsem se rozhodl hledat mezi akcelerometry digitálními. Při výběru akcelerometru bylo pohlíženo především na jeho použité technologie, dostupnost a cenu. Klíčové parametry jsou uvedeny v tabulce.

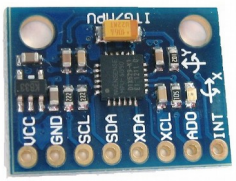
Parametr	Typ zařízení		
	MPU-6050	MMA7455	ADXL345
Vcc (V)	2.37-3.46	2.4-3.6	2.1-3.6
Výstup	Digitální	Digital	Digital
Komunikační rozhraní	I2C	I2C/SPI	I2C/SPI
Rozsah akcelerometru (g)	±16	±8	±16
Počet os	6	3	3
Přidána technologe	gyroskop	žádná	žádná
Velikost (mm)	20x15	25x13	20x14
Cena (Kč)	130	120	160

Tabulka 2.3: Tabulka vybraných typů akcelerometrů

Na základě srovnání těchto 3 typů senzorů jsem vybral akcelerometr firmy Inven Sencse typ MPU-6050 [3]. Tento akcelerometr má oproti ostatním zařízením přidán gyroskop, jeho rozsah je jeden z největších mezi vybíranými akcelerometry a také je cenově přijatelný.

2.5 Akcelerometr MPU-6050

Senzor akcelerometru [3] spolu s gyroskopem je integrován do pouzdra GY-521. Pouzdro je velmi malých rozměrů konkrétně 20x15 mm, proto je tento senzor předurčen k měření hodnot náklonu, zrychlení atd. v RC modelech. Senzor obsahuje celkem 6 os, tedy tři osy akcelerometru X,Y,Z s nastavitelným rozsahem od ±2 g, ±4 g, ±8 g až ±16 g a také tři osy gyroskopu v programovatelném rozsahu uhlového zrychlení od ±250 až po ±2000 °/sec. Senzor se vyznačuje velmi nízkou spotřebou v pohotovostním režimu okolo 5 µA. Mezi velké výhody senzoru patří přidání Digital Motion Processoru (DMP) pomocí tohoto procesoru nemusíme zatěžovat výpočty vlastní mikrokontrolér, ale naopak hodnoty zaznamenané senzorem přenecháme k výpočtům DMP. Tento senzor podporuje typ sběrnice I2C s vlastním I2C kontrolérem, funkce sběrnice bude popsána v kapitole 2.8.2. Vstupě-výstupní obvody jsou připojeny pomocí 8 pinů. Pro naše účely a výpočty nám poslouží piny VCC pro připojení napájecího napětí 3,3 V. Pin GND slouží pro připojení uzemnění. Dále použijeme pro komunikaci pomoci sběrnice I2C následující dva piny a to SDA a SCL. Již takto zapojený akcelerometr je schopen měřit základní hodnoty zrychlení. Avšak chceme-li využít také DMP pro přepočet hodnot, musí se připojit také pin INT, což je pin pro externí přerušení. Detailní zobrazení modulu a jeho pinů je možné vidět na přiloženém obrázku. Toto rozdělení pinů se však může měnit s použitím jiného pouzdra než-li GY-521.



Obrázek 2.3: Senzor MPU-6050

2.6 Mikrokontroléry

Mikrokontrolér označovaný také jako mikropočítač nebo jednočipový mikropočítač je elektronická programovatelná součástka. Můžeme si ji představit jako integrovaný obvod implementován na jediném čipu. Struktura mikrokontroléru bývá zpravidla obsazena takto:

- Procesor (CPU)
- Paměť
- Programovatelné vstupně-výstupní rozhraní
- Přídavní periferní obvody

Takto navržený mikrokontrolér může pracovat ve 2 režimech. Jako mozek nějakého přístroje, kde má funkci řídící nebo jako část přístroje, kde plní určenou specifickou úlohu. Mikrokontroléry se dnes objevují téměř ve všech elektronických přístrojích, jako např. telefony, pevné disky, měřicí přístroje, lékařské přístroje, ale také k řízení různých motorů. V bakalářské práci se zaměřím na mikrokontroléry ATMega od firmy Atmel na open-source (volně dostupné) platformě Arduino.

2.7 Arduino

Arduino [4] je open-source platforma založená na mikrokontrolérech ATMega od firmy Atmel. Tato firma je předním výrobcem polovodičů a integrovaných obvodů od roku 1984. Samotný projekt Arduino vznikl v roce 2005 v Itálii. Cílem tohoto projektu bylo usnadnit práci s mikrokontroléry pro studenty, tím umožnit rychlejší vývoj a zjednodušit ovládání. Tento projekt je již od svého počátků volně dostupný. Pod licencí Creative Commons [4] jsou vydávány příručky jazyka a externích knihoven nebo také dokumentace. Díky tomu je dodržována určitá kompatibilita a je k dispozici obrovské množství zdrojových souborů.

2.7.1 Hardware

Desky Arduino [5] se používají s mikrokontroléry rodiny AVR Atmega [6][7] s RISC architekturou. Používají čipy ATMega8, ATMega168, ATMega328, ATMega1280 a ATMega2560. Mikrokontrolér je uživatelsky programovatelný a obsahuje již bootloader neboli kód, který se po zapnutí stará o veškeré základní nastavení mikrokontroléru. Arduino deska obsahuje také několik diod pro indikaci, resetovací tlačítko, oscilátor, obvod USB obstarávající programování a komunikaci s PC díky softwarově simulované komunikaci přes linku RS-232, napájení 7-12 V, napájecí piny pro ostatní zařízení 3,3 a 5 V. Nedílnou součástí jsou digitální piny pro digitální vstup/výstup a softwarově řízený PWM a také analogové piny s AD převodníkem. K Arduinu je možné připojit rozšiřující desky Shield jako např. Ethernet Shield pro síťovou komunikaci. Existuje několik druhů Arduino desek jako jsou Arduino Uno, Mega 2560, ADK a další. Liší se jak použitými čipy, tak množstvím pinů a velikostí.

V bakalářské práci pracuji se dvěma rozdílnými Arduino deskami a to konkrétně s typem Duemilanove a Mega2560. Jejich porovnání můžete vidět v následující tabulce.

Parametr	Typ desky Arduina	
	Duemilanove	Mega2560
Mikrokontrolér	ATmega328P	ATmega2560
Flash(KiB)	32	256
EEPROM(KiB)	1	4
SRAM(KiB)	2	8
Digitální I/O piny	14	54
PWM kanály	6	14
Analogové vstupy	6	16
Typ USB rozhraní	FTDI	ATmega8U2
Rozměry (mm)	68,6x53,3	101,6x53,3

Tabulka 2.4: Tabulka dvou desek Arduino

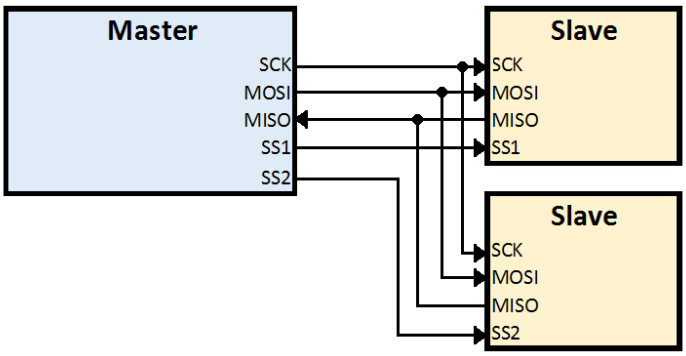
2.8 Sběrnice

Obecně lze sběrnice popsat jako soustavu vodičů sloužící pro přenos signálů mezi jednotlivými periferiemi. Sběrnice mohou přenášet signály datové, adresové, stavové nebo řídící. Signál je možno posílat jednosměrně nebo obousměrně. Přenos dat je možno řídit hodinovým impulsem a vytvořit tak synchronní přenos anebo využít potvrzovací signály, a tím řídit přenos asynchronně. Sběrnice navíc nemusí obsahovat pouze jeden vodič pro sériový přenos, ale také více vodičů pro přenos paralelní. Jedny ze základních parametrů sběrnic jsou: šířka přenosu (bit), frekvence (Hz), rychlost (B/s). V bakalářské práci využívám sběrnice typu SPI a I2C, které jsou součástí mnoha digitálních zařízení, například pro komunikaci s integrovanými obvody a jejich řídícím mikroprocesorem. Jejich hojné využití je dáno především velmi snadnou implementací jak po stránce elektronické, tak softwarové. Další jejich výhodou je schopnost komunikovat s více než dvěma uzly, a také vyvedení samostatného hodinového signálu.

2.8.1 SPI

Sběrnice SPI [8] je navržena pro komunikaci se dvěma nebo více uzly. Princip spočívá v tom, že jeden uzel je veden jako řídící neboli master, ostatní uzly jsou podřízené neboli slave. Uzel v režimu master obsahuje generátor hodinového impulzu, jeho frekvence může dosahovat až 70Mhz. Impulz je rozveden do všech dalších uzlů vodičem a tím je zajištěna synchronní obousměrná komunikace. Tento vodič nese označení SCK (Seriál Clock). Master navíc disponuje dvojicí vodičů označených jako MISO (Master In, Slave Out) jako vstup a MOSI (Master Out, Slave In) jako výstup sloužící pro obousměrný přenos dat. Aby bylo možné vybrat s jakým uzlem slave chce master komunikovat, je zaveden ještě jeden signál a to SS (Slave Select). Uzel slave pokud je aktivován signálem SS vysílá podle hodinového signálu data. Komunikace mezi jednotlivými zařízeními probíhá následovně. Master jakožto řídící uzel

nastaví na zařízení, se kterým chce komunikovat log. 0 pomocí SS signálu a poté na vodiči SCK začne generovat hodinový signál pro synchronizaci vysílání a příjem dat. U výběrového signálu je použit tzv. invertovaný vstup, proto se výběr zařízení slave provádí log. 0 nikoli 1. V tuto chvíli vyšlou obě zařízení svá data, master je vyšle po vodiči MOSI a zároveň slave po vodiči MISO. Komunikace může dále pokračovat pokud master stále dodává SCK signál a hodnotu SS nezměnil, nebo může být komunikace ukončena změnou SS signálu do log. 1 a zastavením hodinového signálu.



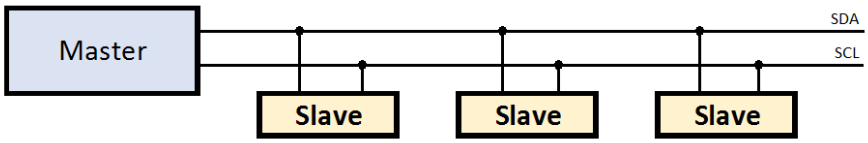
Obrázek 2.4: Schema SPI sběrnice

Ve většině zařízení je možnost nastavení polarity hodin a také, kdy má dojít k vysílání a přijímání dat, zda na vzestupné či sestupné hraně hodinového signálu. Proto jsou k dispozici 4 kombinace nastavení polarity a fáze hodin. Ty se nastavuji dvěma bity. Pro nastavení polarity hodinového signálu slouží konfigurační bit CPOL. Tímto bitem se nastaví klidová úroveň neboli stav, kdy nedochází k vysílání dat. Druhým bitem CPHA se nastavuje, zda se data budou vysílat nebo číst po příchodu vzestupné či sestupné hrany. Mezi velké výhody takto navržené sběrnice patří především její jednoduchost. Jednoduché je jak hardwarové rozhraní sběrnice, které ve své podstatě nepředstavuje nic jiného než několik propojených posuvných registrů, kde posuv je řízen hodinovým signálem, tak jednoduchý protokol přenosu. Ten je díky dvěma vodičům obousměrný bez nutnosti přepínání mezi příjmem a vysíláním. Hlavní nevýhoda této sběrnice je existence pouze jednoho řídícího zařízení, a také nutnost použití čtyř vodičů. Tato nevýhoda je částečně kompenzována používáním tohoto typu sběrnice pouze na krátké vzdálenosti.

2.8.2 I2C

Sběrnice I2C [9] má některé prvky společné jako sběrnice SPI. Jednou ze společných věcí je rozdělení uzlů opět na master a slave. I v tomto případě bývá jako uzel master zvoleno řídící zařízení např. mikrokontrolér, v jeden okamžik může existovat pouze jedno zařízení typu master. Zbylé uzly pracující v režimu slave a nemohou do řízení sběrnice nijak zasahovat ani posílat žádosti o příjem či vysílání dat. Další společnou věcí je použití hodinového signálu a tím synchronního přenosu dat. Ovšem na rozdíl od sběrnice SPI, se jedná pouze o poloduplexní přenos, tedy v jednu chvíli může existovat jedno zařízení vysílající a jedno zařízení přijímající. Sběrnice I2C nedisponuje signálem SS určeným pro výběr slave zařízení. Zde má každý slave

přiřazenou jednoznačnou adresu. Tím se dostáváme k hlavnímu rozdílu mezi sběrnici SPI a I2C a to, že sběrnice I2C obsahuje také základní komunikační protokol, o kterém se budu zmiňovat v textu níže. Celá sběrnice je tedy tvořena dvojicí signálových vodičů. První signálový vodič SDA (Serial Data) slouží pro přenos dat. Přes druhý signálový vodič SCK (Serial Clock) posílá zařízení master hodinový signál všem ostatním zařízením. Tyto dva vodiče je nutné připojit přes rezistory o velikosti odporu 1,5 kΩ na napájecí napětí a to z důvodu, že během nečinnosti všech uzlů zvednou odpory logickou úroveň obou signálových vodičích na 1. Tento stav je normovaný a představuje klidovou úroveň.



Obrázek 2.5: Zapojení sběrnice I2C

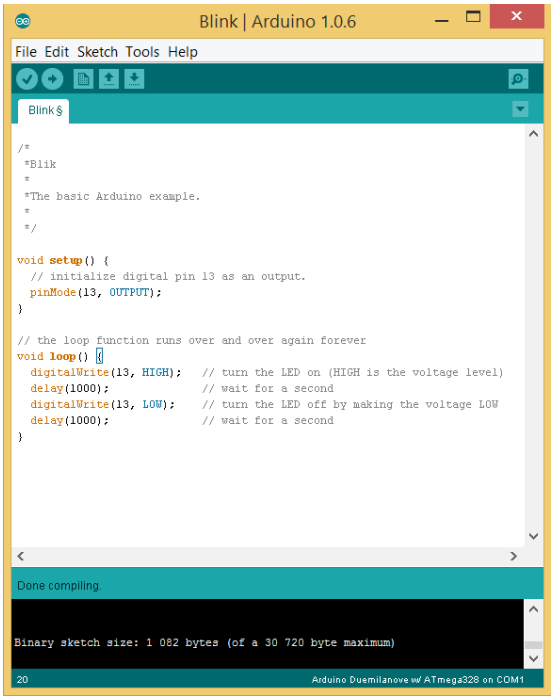
Řízení celé sběrnice provádí zařízení master. Zahájení komunikace probíhá, pokud nejsou přenášena žádná data a napěťové úrovně obou signálových vodičů SDA a SCL jsou na log. 1. Master sníží na vodiči SDA logickou úroveň na 0, naopak na vodiči SCL udržuje určitou dobu log. 1 v závislosti na zvolené přenosové rychlosti, poté také na tomto vodiči přepne logickou hodnotu na 0. Tento stav se nazývá start bit a týká se všech připojených uzlů. Jako první po vyslání start bitu začne master posílat adresu uzlu, se kterým chce komunikovat. Tato adresa může být sedmibitová, tím pádem lze adresovat až 128 různých zařízení, nebo desetibitová a ta podporuje adres 1024. Prakticky je však tento počet adres u obou verzí menší z důvodu rezervování určitých adres. Sedmibitová adresa je přenášená v jednom bytu, kde poslední bit s nejnižší váhou představuje přepínač mezi módem čtení nebo zápisem dat. Jakmile je těchto 8 bitů odesláno, všechny zařízení slave testují, zda se odeslána adresa shoduje s jejich vlastní. Pokud adresa souhlasí slave vyšle potvrzovací bit ACK (Acknowledge) jako log. 0 zpět zařízení master a dá tím najevo, že je připravena pro komunikaci a dojde k zahájení transferu dat. Každý přijatý byte je potvrzen bitem ACK stejně jako v případě zahájení komunikace. Desetibitová adresa se posílá ve 2 bytech, přičemž první byte odpovídá vzoru 11110XX. Poslední bit musí končit vždy 0 z důvodu, aby slave přijal druhý byte adresy. Dále komunikace probíhá stejně jako v případě sedmibitové adresy. Sběrnice I2C má několik standardizovaných rychlosti a to 10 kbps, 100 kbps a 400 kbps. Oproti sběrnici SPI je I2C sběrnice o něco složitější, ale zase má výhodu v použití na větší komunikační vzdálenosti v řádech metrů.

3. Softwarová implementace

Jakmile vybereme vhodný typ senzoru, který nám bude sloužit jako zdroj dat pro náš protokol, k němu vybereme vysílače a přijímače sloužící pro bezdrátový přenos a mikrokontroléry starající se o celkový chod celého zařízení, je na řadě začít implementovat programy zajišťující správnou funkčnost všech součástek. Zaměříme se zde na implementaci programu pro akcelerometr, přijímač a vysílač. Hlavní část této kapitoly bude věnována návrhu komunikačního protokolu spolu s programem pro zobrazování hodnot.

3.1 Vývojové prostředí Arduino

Aby byla platforma určená především studentům a domácím kutilům úspěšná, byl kladen velký důraz na jednoduché vývojové prostředí. A proto vzniklo prostředí Arduino IDE (Integrated Development Environment) [10]. Toto prostředí obsahuje textový editor s tlačítky, které mají danou jasnou funkci tak, aby byl každý schopen psát a nahrávat vlastní programy do Arduina. Aby bylo docíleno multiplatformnosti je Arduino IDE napsáno v jazyce Java. Proto stačí, aby operační systém, na kterém budeme programovat obsahoval Java Virtual Machine. Rozhraní vývojového prostředí nese původ v nástroji Wiring, což je programovací jazyk. Vývojové prostředí je spojené s projektem Processing neboli jazykem a IDE vyvinutém pro výuku programovaní. Wiring je velmi podobný jazyku C a C++ liší se jen minimálně, většinou se jedná o zjednodušení a malé modifikace. Základní struktura programu s povinnými bloky a celé Arduino IDE je vidět na obrázku.



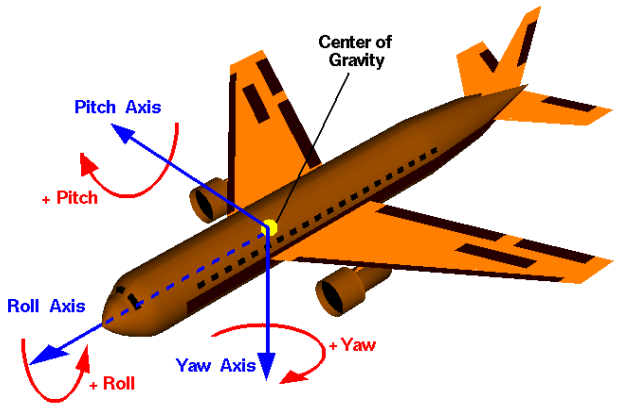
Obrázek 3.1: Vývojové prostředí Arduino IDE

Celý kód programu se skládá z inicializační části a dvou bloků. V inicializační části se nastavují knihovny, konstanty, proměnné, mohou se zde vytvářet funkce atd. Za inicializační části následují bloky setup a blook loop. V prvním bloku se mezi složené závorky napíše kód, který se spustí pouze jednou a to na začátku programu. Začátkem programu se rozumí připojení Arduino desky k napájení, restartování pomocí tlačítka reset, nebo nahrání kódu do Arduina. Ve druhém bloku se mezi složené závorky zapíše kód, který se bude opakovat neustále dokola, dokud nedojde k odpojení Arduina. Toto jsou dva základní bloky programu a musí být v kódu pokaždé bez ohledu na to, zda obsahuji či neobsahují příkazy. K vytvoření složitějších projektů ovšem budeme potřebovat předdefinované knihovny neboli libraries. Knihovny jsou alfou a omegou celého projektu Arduino. Existuje obrovské množství různých druhů knihoven pro nejrůznější druhy senzorů, shieldů a modulů, které nejsou ve standardní výbavě Arduino IDE. Knihovny jsou dostupné volně na internetu. Po stažení se jednoduše vloží do složky libraries a po restartování Arduino IDE je můžeme nalézt v menu Sketch » Import Library. Velké množství knihoven obsahuje také řadu ukázkových kódů, ty jsou dostupné v menu File » Examples. Pokud máme naimportovány knihovny a napsaný kód, můžeme ho zkontrolovat pomocí tlačítka verify, tím dojde ke kontrole chyb, ty jsou případně vypsány ve spodní části IDE spolu s velikostí celého projektu. Takový projekt se ukládá do námi určeného adresáře neboli sketchbooku. V něm se vytvoří podsložka se stejným názvem jako je jméno projektu a tato podsložka obsahuje zdrojové soubory nesoucí příponu .ino nebo .ide. Pokud chceme program nahrát na naší Arduino desku musíme nejprve nastavit správný druh desky, nastavení provedeme kliknutím na název naší desky v sekci Tools » Board. Dále musíme vybrat k jakému sériovému portu je Arduino připojeno. Nastavení sériového portu provedeme v sekci Tools » Serial Port. U Mac počítačů bude sériový port podobný /dev/tty.usbmodem241 nebo /dev/tty.usbserial-1B1. U Linux systému /dev/ttyUSB0 a /dev/ttyUSB1 a v mém případě, což je Windows se jedná o porty COM1, COM2 a další. Pokud máme správně nastaveny všechny parametry a správně zkompilovaný program, můžeme kliknout na tlačítko Upload. Po zmáčknutí se automaticky restartuje připojená Arduino deska a začne nahrávání. K nahrávání slouží takzvaný bootloader, tedy program nahraný v desce starající se o upload programu bez použití dalšího hardwaru. Po dokončení uploadu se rozblikají LED diody TX a RX a IDE prostředí vypíše, zda došlo ke správnému nahrání programu, či k chybě. Program v Arduino desce od této chvíle běží neustále, dokud je deska připojena k napájecímu napětí.

3.2 Software pro akcelerometr

Zařízeni akcelerometr spolu s gyroskopem, které jsem zvolil, je velmi vhodný typ senzoru, díky němu si můžeme vyzkoušet jak programování Digital Motion Processoru (DMP), ten je osazen přímo na našem zařízení, tak propojení s komunikačním protokolem, který pracuje s hodnotami získanými právě díky DMP. Pro práci se zařízením MPU-6050 existuje knihovna vytvořena Jeffem Rowbergem s názvem MPU-6050 [11]. Tato knihovna umožňuje práci se zařízením jak formou čistého získávání dat, tak výpočet a získání hodnot náklonu pomoci DMP. Pokud chceme získat pouze surová data z akcelerometru a gyroskopu, musíme v první řadě připojit zařízení ke sběrnici I2C, k tomu využijeme knihovnu Wire a I2Cdev, poté inicializujeme

UART komunikaci. K získání hodnot nám slouží metody getAcceleratino a getRotation. Veškeré nastavení akcelerometru a gyroskopu jsou ve výchozím stavu, proto je možno výsledné hodnoty upravit změnou citlivosti v metodách setFullScaleGyroRange a setFullScaleAccelRange. Nastavitelné uhlové zrychlení pro gyroskop je v rozsahu ±250 °/sec, ±500 °/sec, ±1000 °/sec, ±2000 °/sec. Nastavení přetížení u akcelerometru ±2 g, ±4 g, ±8 g, ±16 g. Získané hodnoty vypadají takto: *ax:616 ay:1516 az:16188 gx:1849, gy:-698, gz:-270*. Vypovídající informace takto získaných hodnot však není velká. Proto je lepší použití DMP, ten dokáže provádět výpočty s naměřenými hodnotami a tím nepřenáší zátěž na řídící jednotku Arduino. Vzhledem k tomu, že s jakýmkoliv pohybem senzoru se do hodnot promítne také setrvačnost, je potřeba k získání stabilních uhlů, ve kterém se nacházíme velké množství výpočtů. V knihovně MPU- 6050 se tyto výpočty nacházejí a proto je pouze nutné úhly získat včas díky přerušení. Nastavíme si tedy obsluhu DMP, ta nám bude vyvolávat přerušení změnou proměnné false za true, jakmile jsou data k dispozici. Pokud dojde k přerušení, zjistíme si velikost paketu připraveného v DMP. Do předem připraveného FIFO zásobníku si uložíme data. Poté si pomocí získané velikosti paketu z DMP určíme jaká část dat se má ze zásobníku načíst a použít pro výpočet. Naším cílem je získání úhlů, které se používají u letadel a koptér. Jsou to úhly yaw, pitch a roll zobrazeny na obrázku.



Obrázek 3.2: Úhly Y, P, R

Aby bylo možné vůbec tyto uhly získat, musíme nejprve vypočítat tzv. kvaternion, k tomu slouží metoda dmpGetQuaternion, poté si musíme z hodnot kvaternionu vypočítat vektor setrvačnosti pomocí metody dmpGetravity. Nyní nám už zbývá vypočítat úhly metodou dmpGetYawPitchRoll. Takto získané hodnoty poté jednoduše převedeme z radiánu na stupně, uložíme je a můžeme je vypsat na sériový port: *Y:-169.34 P:51.40 R:-4.06, Y:-168.96 P:48.65 R:-2.35*

3.3 Software pro transceiver NRF24l01

Pro transceiver NRF24l01 existuje celá řada knihoven pracující s platformou Arduino. Mezi jedny z nejpoužívanější patří RF24, Mirf, AVRlib, RadioHead. Po prozkoumání všech knihoven jsem si nakonec vybral RF24. Tato knihovna byla z mého pohledu nejpřehlednější, byla snadno implementovatelná, její metody umožňovaly celou řadu nastavení jak vysílače, tak přijímače a jeden z hlavních důvodů, proč jsem zvolil tuto knihovnu je, že na základě této knihovny je vytvořena knihovna RF24Network. Její porovnání s technologii ZigBee můžete vidět v kapitole 1. RF24Network se svými možnostmi nejvíce přibližuje mé představě při práci s bezdrátovým přenosem. Tato knihovna byla navržena panem Jamesem Colizem v roce 2011 jako alternativa právě k technologii ZigBee [12]. Tato knihovna se velmi hodí do konceptu mé bakalářské práce, kdy vytvářím alternativu k již zaběhlým způsobům komunikace. Knihovna RF24Network tedy vychází z knihovny RF24, poskytuje nám hostitelské adresování, kde každý uzel má logickou adresu na lokální síti, dále umožňuje přesměrování zpráv, díky čemuž mohou být zaslány zprávy z jednoho uzlu do jakéhokoliv jiného. Podporuje možnost Ad-hoc připojení, kdy se uzel připojí do sítě, aniž by ovlivnil všechny existující uzly. Uspořádání sítě je do stromové topologie, tedy jeden uzel je základní a všechny ostatní uzly jsou „děti“. Je zde využíváno oktalové řešení adres a každý uzel musí mít přiřazenou patnácti bitovou adresu správce. Každé další číslo v adrese reprezentuje pozici uzlu ve stromu dále o základny. Naše transceivery jsou připojeny k mikrokontroléru přes sběrnici SPI. Proto pro správnou funkci programu musíme implementovat nejen knihovnu RF24 spolu s navazující knihovnou RF24Network, ale také knihovnu SPI. Po includování všech knihoven přichází na řadu připojení NRF24l01 na sběrnici SPI. Pro připojení ke sběrnici SPI jsem použil piny 9 a 10, ty odpovídají Arduino desce Duemilanove, pro desku MEGA2560 jsou to piny 40 a 53. V obou případech se jedná o piny digitální a v závislosti na typu desky se mohou měnit. Jakmile jsou Arduino desky připojeny ke sběrnicím, provedeme adresaci. Vytvořil jsem spojení, kdy jeden z transceiveru bude pracovat jako vysílač, jeho logická adresa je 00 a druhý bude pracovat jako přijímač s logickou adresou 01. Veškeré ostatní nastavení se provádí ještě před samotnou částí setup. V té poté dojde k nastavení baut rate, přenesení připojení a vytvoření RF kanálu s hlavním uzlem pro komunikaci.

Po tomto nastavení se může blok setup ukončit. Následuje smyčka loop, kde na samém začátku musíme zavolat metodu update. V místě volání této metody dochází vždy ke všem změnám během procesu a kontrole sítě. Za touto metodou je možné dále nastavit např. interval posílání zpráv, indikaci odeslání paketu pomocí LED diody, výpis systémových hlášení atd.

Nyní se dostáváme k samotnému odesílání dat. Chceme-li odeslat naše data, nejprve musíme vytvořit hlavičku paketu, k tomu nám slouží konstruktor RF24NetworkHeader, ten nám vytvoří hlavičku v daném formátu.

4B	4B	4B	1B	1B	4B
from_node	**to_node**	**id**	**type**	**reserved**	**next_id**

Obrázek 3.3: Struktura hlavičky paketu

Struktura hlavičky paketu neboli header má celkem 5 proměnných:

- From_node: Logická adresa, kde se zpráva vytvoří.
- To_node: Logická adresa, na kterou bude zpráva odeslána.
- Id: Id zprávy, zvyšující se s každou novou zprávou.
- Type: Typ paketu, 0-127 uživatelsky definovaný, 128-255 vyhrazen pro systém.
- Reserved: Vyhrazeno pro budoucí využití.
- Next_id. Id zprávy, která bude následovat.

Takto vytvořená hlavička obsahuje námi zadané parametry from_node: 00, to_node: 01, id: 1, typ je defaultně nastaven na 0, rezerved je nevyužita a next_id se vždy počítá jako next_id++. Po vytvoření paketu následuje již samotné odeslání zprávy druhému zařízení.

Odeslání dat se provádí metodou write. Tato metoda má 3 parametry a to hlavičku header, kterou jsme si předem vytvořili, ukazatel na místo v paměti, kde máme uložená data ze senzoru a jako třetí parametr je zde velikost dat určených pro odeslání. Druhý parametr bude detailněji popsán v kapitole 3.4. Program pro odesílání dat můžeme ukončit a nahrát soubor do Arduino desky, na kterou jsou připojeny senzory spolu s transceiverem pracujícím jako vysílač. Program pro zařízení přijímající data bude velice podobný. Určíme si podle typu desky piny pro komunikaci přes SPI, nastavíme si logické adresy, v bloku setup také nastavíme baut rate, přenesení připojení a vytvoříme RF kanál s hlavním uzlem pro komunikaci. Ve smyčce loop nemůžeme zapomenout na metodu update pro pravidelnou kontrolu sítě a aktualizaci. Poté si opět vytvoříme pomocí konstruktoru hlavičku. Nyní využijeme cyklu while a metody available k tomu, abychom ověřovali, zda je připravená nějaká zpráva pro náš uzel. Jakmile je zpráva detekována můžeme ji přečíst.

Čtení paketu se provádí metodou read. Metoda má 3 parametry. První parametr je hlavička přijaté zprávy, druhý je ukazatel na paměť, kde by měla být zpráva uložena a třetí parametr je největší velikost přijaté zprávy. Druhý parametr bude rozebrán podrobněji v kapitole 3.4. Metoda nám vrací celkový počet bytu zprávy. Na závěr si přečtená data spolu s velikosti zprávy odešleme na sériový port. Takto vytvořený program nahrajeme do Arduino desky, ke které je připojen transceiver a ten bude pracovat jako přijímač

3.4 Návrh obecné uživatelské části paketu

Jedním z úkolů bakalářské práce je navrhnout způsob, jak posílat data z použitých senzorů v obecném formátu. Adresová část paketu neboli header popsána v kapitole 3.3 je pevně daná. Uživatelská část paketu je reprezentována ukazatelem message a je uložena v paměti. Tato část obsahuje uživatelská data poskytnuta senzory spolu s dalšími proměnnými nezbytnými ke správnému chodu, rozpoznání chyb a zpracování přenosu. Má pevně daný formát, na jehož začátku je určen start byte. Start byte bude začínat vždy stejnou předem určenou hexadecimální hodnotou a to z důvodu, aby bylo možné rozpoznat začátek uživatelské části paketu. Vzhledem k tomu, že můžeme k našemu řídícímu zařízení, což je v našem případe Arduino Mega2560 připojit několik různých senzorů, jsem zvolil jako druhou proměnou uživatelské část id senzoru. Každému senzoru připojenému k řídícímu zařízení se přiřadí unikátní id. Toto id slouží jako rozeznávací prvek. Třetí částí je celková velikost dat získaných ze senzoru. Následují samotná data. Čtvrtou položku jsem volil s ohledem na to, že během přenosu může dojít k několika chybám, proto jsem použil CRC (Cyclic redundancy check) neboli kontrolní redundantní součet sloužící k detekci chyb. O této hašovací funkci se budu zmiňovat v kapitole 3.4.1, bude zde popsána jak samotná funkce, tak její implementace. Poslední proměnnou uživatelské části paketu je end byte. Tento end byte bude podobně jako start byte konstantní hexadecimální hodnoty a bude představovat konec uživatelské části paketu.

1B	1B	1B	data_len	1B	1B
start_byte	**id_senzor**	**data_len**	**data**	**CRC**	**end_byte**

Obrázek 3.4: Formát uživatelské části paketu

Takto vypadá struktura uživatelské části paketu. Obsahuje celkem 6 proměnných:

- Start_byte: Tímto bytem začíná uživatelská část paketu a má hodnotu 0xAA.
- Id_senzor: V našem případě je id akcelerometru 1 a id teploměru 2.
- Data_len: Délka datové části se liší v závislosti na použitém senzoru a jeho výsledných hodnotách. U akcelerometru pracujeme se 3 proměnnými typu float, proto je hodnota proměnné data_len pro akcelerometr 12B (3 x 4 B). U teploměru počítáme s dvěma veličinami typu float, tomu odpovídá hodnota data_len 8 B (2 x 4 B).
- Data: Naměřené a zpracované hodnoty ze senzorů. Maximální velikost dat je 27 B.
- CRC: Osmibitové CRC určeno k detekci chyb.
- End_byte: Byte znázorňující konec uživatelské části paketu má hodnotu 0xBB.

Uživatelská část paketu může mít maximální velikost 32 B, 5 B je vyhrazeno pro nedatové proměnné, na samotná data nám tedy zbývá předem zmiňovaných 27 B. Pro většinu senzorů je takováto velikost dostačující, avšak pro GPS senzor poskytující data ve formě RMC (Recommended Minimum Navigation Information), neboli větě nesoucí minimální doporučenou informaci o navigaci je už velikost 27 B nedostačující.

Implementace uživatelské části programu se provádí v kódu spolu se zpracováním hodnot ze senzorů a odesláním, či příjmem dat. Nadefinují se start a end byty. Poté se pro jednotlivé druhy senzorů nastaví jejich id spolu s délkou datové části. Implementaci prvních třech a posledního bytu se provádí v inicializační části programu. Obdobně by probíhalo nadefinovaní počátečních hodnot pro všechny typy senzorů, které bychom připojili k Arduino desce. Jakmile známe délku dat, vytvoříme si pomocnou proměnou buffLenght. Tato proměnná nám bude představovat celkovou velikost uživatelské části paketu. Nyní vytvoříme pomocí dynamické alokace paměti pole bytu, které bude mít velikost buffLenght. Následně do takto vytvořeného pole uložíme na správná místa proměnné start byte, id senzoru a velikost dat. Pro samotné uložení dat si zjistíme ukazatel na první byte v paměti, kde se má nacházet naše první datová proměnná, v našem případě se jedná o úhel Y z akcelerometru, začínající na pozici s indexem 3. Poté si ukazatel byte přetypujeme na ukazatel float, protože naše hodnota Y je typu float a operátorem dereference uložíme hodnotu z akcelerometru. Stejně postupujeme pro všechny datové proměnné ze senzoru. Nyní si spočítáme a uložíme CRC na předposlední byte uživatelské části a nakonec uložíme samotný ukončovací znak end byte.

```
char buffLength = ACC_DATA_LEN + 5;
char* buff = new char[buffLength];
buff[0] = START_BYTE;
buff[1] = ACC_ID;
buff[2] = ACC_DATA_LEN;
*((float*)&buff[3]) = ypr[0] * 180 / M_PI;
*((float*)&buff[7]) = ypr[1] * 180 / M_PI;
*((float*)&buff[11]) = ypr[2] * 180 / M_PI;
buff[buffLength - 2] = getCRC(buff, 1, buffLength - 2);
buff[buffLength - 1] = END_BYTE;
```

Výpis 1: Inicializace uživatelské části paketu na Arduino desce

Po uložení odešleme data příjemci a musíme smazat vytvořené pole destruktorem delete, jinak by došlo k přeplnění paměti a pádu Arduino desky. Tento příklad vytvoření a uložení pole byl uveden pro senzor akcelerometr.

3.4.1 CRC

Cyklický redundantní součet je funkce, používána k detekci chyb během přenosu dat, ale také během jejich ukládání [13]. Jedná se o velmi rozšířený způsob kontroly správnosti dat především pro svou jednoduchost a dobré vlastnosti. Princip kontrolních součtů spočívá v tom, že se z dostupných dat pomocí funkce se zvoleným klíčem vypočítá výsledek, ten se připojí k datům a odešle se příjemci. Příjemce si poté přijatá data přepočítá se stejným klíčem a výsledek přepočítaných dat porovná s výsledkem přijatým. Pokud si jsou porovnané výsledky rovny, s největší pravděpodobností nedošlo k chybě. Pravděpodobnost odhalení chyby roste s šířkou zvoleného klíče. Podoba klíče se většinou určuje podle druhu přenosu, klíčem se dá ovlivnit, jaký typ chyb bude snáz odhalen.

V našem případě se samotný výpočet kontrolního součtu provádí ve funkci getCRC, kde si jako první určíme klíč, ten má osmibitovou hodnotu 0x91 neboli 10001001. Následuje samotná funkce getCRC se třemi parametry. První parametr je message což představuje ukazatel na místo v paměti, kde je uložena naše uživatelská část paketu. Následují parametry from a to, určující začátek a konec dat, které mají být použit pro výpočet CRC. Tyto dva parametry jsou zde proto, že při výpočtu CRC se ignoruje start, end a CRC byte. Následuje doplnění sedmi nul na konec výpočtu, kde jeden bit je vynechán z důvodu nastavení CRC. Poté dojde k samotnému výpočtu. Klíč se zarovná pod daty určenými k výpočtu. Pokud je klíč zarovnán pod logickou 1 dat, provede se operace XOR se zvoleným klíčem, výsledek se uloží a posune se ukazatel o jeden bit vpravo. Jakmile je zarovnán pod logickou 0, operace XOR se neprovádí. Jakmile je takto přepočteno všech 8 bitů, ukazatel se přesune na další byte uživatelské části paketu. Výsledek funkce je poté zbytek po výpočtu všech dostupných dat. Toto je naše hodnota CRC a ta se uloží do paketu. Existuje však i druhá možnost jak spočítat hodnotu CRC a to hardwarově. Naše transceivery podporují hardwarovou kontrolu dat a to osmibitovým a šestnáctibitovým CRC. Bohužel naše knihovna však s takovou kontrolou CRC nepočítá, a proto je kontrola chyb řešena softwarově. Kontrola se provádí jak na straně příjemce, tak následně v samotné uživatelské aplikaci. Dochází tedy k dvojí kontrole dat, což má za následek zobrazení menšího množství potencionálně chybných paketů.

3.5 Uživatelská aplikace

Arduino deska je pomocí USB kabelu připojena k počítači. Data přijatá Arduino deskou jsou odeslána na sériový port. Naše uživatelská aplikace bude mít za úkol číst data ze sériového portu, zpracovat do požadovaného formátu a následně je uživateli určitou formou zobrazit. Aplikace je vytvořena v jazyce C#. V tomto jazyce mi přišlo snazší vytvořit uživatelské rozhraní než v jazyce C++. Celá aplikace se skládá ze 3 projektů: SensorConsole, SensorReader a SensorUI.

3.5.1 SensorReader

Projekt SensorReader představuje knihovnu, ta se sestavuje do dll souboru a obsahuje 4 třídy: SensorData, AccelerometerData, ThermoMeterData a Communication. Třída SensorData nám bude představovat způsob, jakým si uchováme data přijatá z Arduino desky a dochází v ní k "rozparsování" datového pole na jednotlivé části jako id, data a CRC. Obsahuje také metodu ToString pro výpis těchto části na obrazovku. Tato třída slouží jako předek pro třídy AccelerometerData a ThermoMeterData, které z ní dědí.

Třída AccelerometerData obsahuje 2 konstruktory. Prvnímu konstruktoru předávám data, délku dat a CRC. Druhému konstruktoru se předá objekt třídy SensorData a využívá se chceme-li přijatá SensorData převést na AccelerometerData. Uložíme si id senzoru. Následují vlastnosti v podobě úhlů Y, P, R. Ty si vytvoříme vytažením správného množství bytu z dat a převedením na požadovaný datový typ, v našem případě se jedná o 4 byty převedené na float. Třída také obsahuje metodu pro výpis uhlů spolu s id senzoru. Totéž obsahuje třída ThermoMeterData s tím rozdílem, že místo vlastností Y, P, R pracujeme s teplotou a vlhkostí.

Dostavíme se k třídě Communication ta představuje nějakou nadstavbu nad sériovým portem, kde se aplikuje protokol naprogramovaný na Arduino desce. Třída Communication dědí ze třídy SerialPort. V jejím konstruktoru se poté zavolá base konstruktor a předá parametry jako port COM, baud rate, paritu, data byty a stop byty a nastavuji si jakékoli členy třídy nad rámec sériového portu. V našem případě se jedná o příznak reading a reader vlákno, které symbolizuje metodu Run. Ještě před touto metodou se nachází metoda Start, ta otevře sériový port a spustí reader vlákno. Na metodu start navazuje metoda Stop, ta uzavře port a dá vláknu reader 500ms na ukončení. Nyní k samotné metodě Run, využívá se v ní příznaku reading jako podmínky pro cyklus while. Jakmile se tedy reading nastaví na hodnotu false, dojde k ukončení vlákna. Vytvoří se buffer velikosti 280 bytu, velikost bufferu je lehce nadsazena, aby nešlo k přetečení a pádu programu. Do pomocné proměnné načtu první byte a porovnám, zda se jedná o znak 0xAA, pokud není, čtu dál. Jakmile narazím na znak 0xAA, uložím ho a pokračuji na další byte, což je id senzoru. Id si uložím a následuje počet bytu, které se mají přečíst, ty uložím také a na základě této hodnoty přečtu potřebné množství bytu jako data. Nyní zbývá už jen přečíst přijaté CRC a uložit poslední byte. Na základě přijatých hodnot poté spočítám znova CRC stejným způsobem jako v programu na Arduino desce a uložím. K výpočtu CRC je v této třídě určená funkce ComputeCRC. Ze všech předchozích načtených dat ze sériového portu vytvořím SensorData objekt. Toto je první část metody Run. Ve druhé části si zjistím, zda vypočítané CRC bylo správné. Jakmile CRC bylo chybné, tak pomocí podmínky zjišťuji, zda naslouchají události ErrorRaised nějaké funkce, pokud ano, vyvolá se událost a dojde ke spuštění funkcí a předají se jim parametry konkrétně pro výpis chybového hlášení. Obdobně testuji, zda byl přečten správný stop byte, pokud ne, opět zjišťuji, zda naslouchají nějaké funkce události ErrorRaised a v případě že ano, vyvolá se událost a dojde ke spuštění funkcí, v našem případě k výpisu chybového hlášení. Jakmile jsou všechny hodnoty v pořádku a zároveň události SensorDataReceived naslouchají funkce, tak se na základě id senzoru pomocí přepínače switch

vytvoří konkrétní objekt pro daný senzor a zavolá se událost SensorDataReceived, která spustí funkci pro výpis hodnot senzoru na sériový port.

Třída Communication obsahuje také výčtový typ Error s výčtem konstant pro chybné CRC a stop byte. Součástí třídy jsou také dva delegáti a dvě události, ty se využívají právě v metodě Run.

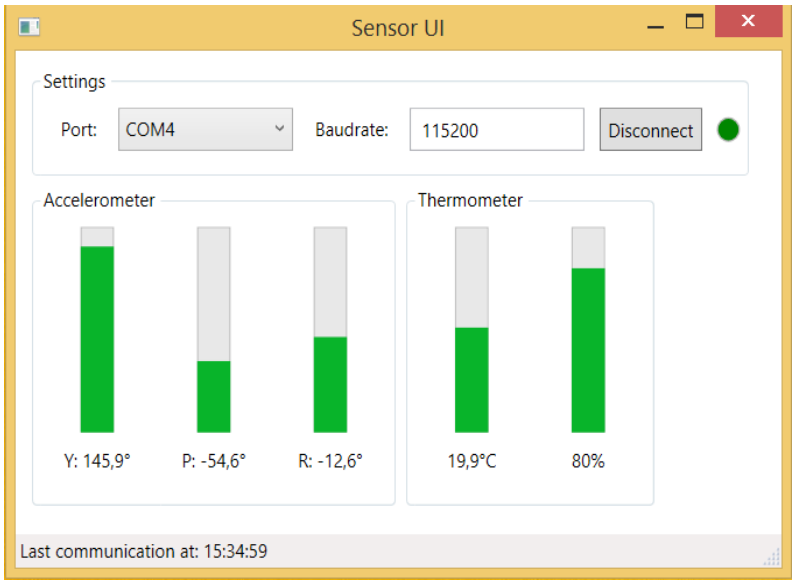
3.5.2 SensorConsole

Projekt SensorConsole obsahuje hlavní metodu Main. V ní si poté vytvořím komunikaci, události SensorDataReceived přiřadím funkce stejně jako události ErrorRaised. Nastartuji si komunikaci metodou Start, a poté metodou Join čekám na konec vlákna. Poslední dvě funkce poté slouží pro výpis error zpráv a dat na sériový port a představují funkce přiřazené událostem zmíněným výše.

3.5.3 SensorUI

Jedním z úkolů bakalářské práce je také zobrazení přijatých hodnot jinou formou, něž prostým výpisem na konzoli. Tím se dostanu do další části a tou je samotné uživatelské rozhraní. Pro tvorbu formuláře jsem použil framework WPF (Windows Presentation Foundation). Důvod proč jsem si zvolil zrovna tento framework je, že v dnešní době se nové aplikace nevytváří ve starším WF (Windows Forms), ale v technologicky vyspělejším WPF. Při práci s návrhem samotného formuláře budu pracovat s jazykem XAML.

V našem WPF projektu je vytvořeno hlavní okno MainWindow. V samotném okně je vytvořena tabulka Grid, do níž jsou poté přidávány následující kontrolky. V grafickém návrháři je pomocí kontrolky GroupBox vytvořeno základní nastavení Settings. V něm je vytvořen ComboBox, ve kterém se nastaví při počáteční inicializaci jednotlivé volby COM portů, jež jsou k dispozici na daném počítači. Dále Settings obsahuje TextBox, v němž se nastavuje baud rate shodný s hodnotnou na Arduino desce. Poté následuje samotné tlačítko pro spuštění a ukončení přenosu Connect a signalizační kolečko. Signalizační kolečko nám bude podle barev zobrazovat stav přenosu. Po kontrolce Settings následují samotné prvky pro jednotlivé typy senzorů. V našem případě se jedná o prvky AcceleroMeter a ThermoMeter. Prvek AcceleroMeter je opět založen na tabulce Grid do níž jsou přidány tři ProgressBary reprezentující úhly Y, P, R. Ke každému ProgresBaru je přiřazeno textové pole, které slouží pro výpis úhlu ve stupních. Obdobně je navržen prvek ThermoMeter s tím rozdílem, že místo třech ProgresBaru jsou použity pouze dva a to pro zobrazení teploty a vlhkosti. I k těmto ProgressBarum jsou přiřazeny textové pole vypisující hodnoty teploty ve stupních Celsia a vlhkosti v procentech. Úplně ve spodní části hlavního okna je umístěn StatusBar zobrazující čas poslední komunikace.



Obrázek 3.5: Uživatelské rozhraní aplikace

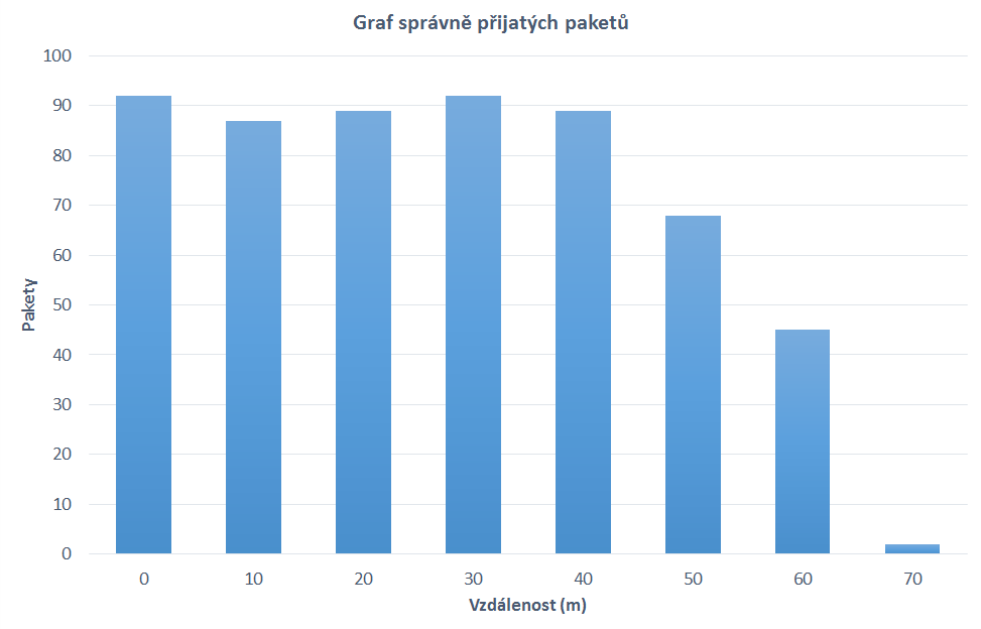
Po prezentační části se dostáváme do tzv. Code Behind, což představuje kód na pozadí, obsahující volaní logiky naší aplikace. Jako první si představíme logiku u třídy AcceleroMeter. V této třídě dojde k nastavení mezních hodnot pro jednotlivé ProgressBary spolu s uložením výchozích hodnot. Třída obsahuje také vlastnosti Y, P, R, kterými nastavíme aktuální hodnoty přijaté ze senzoru a zformátujeme text pro výpis úhlů ve stupních. Třída ThermoMeter je obdobná s tím, že v ní pracujeme s vlastnostmi Temperature a Humidity.

Nyní se dostáváme ke třídě MainWindow. Ta obstarává načítání všech portů COM dostupných na počítači a následné přidání do ComboBoxu určenému pro výběr portů. Dále třída obsahuje cyklickou frontu, do níž jsou ukládány barvy, které se mají zobrazit na signalizačním kolečku indikujícím stav přenosu. Samotnou indikaci má v této třídě na starosti vlákno. To vytahuje barvy z předem připravené fronty a nastavuje je jako výplň pro signalizační kolečko. Třída také implementuje metodu Button_Click pro obsluhu tlačítka Connect. Po zmáčknutí tlačítka metoda přetypuje zadaný baud rate z TextBoxu na datový typ integer. Jakmile přetypování selže, vypíše se chybové hlášení spolu s nastavením barvy pro signalizační kolečko znamenající chybu. Jakmile je zadaný baud rate správný, vytvoří se komunikace jako objekt třídy Communication, spustí se přenos a text tlačítka Connect se změní na Disconnect. Jakmile jsou přijatá data senzoru k dispozici, zavolá se funkce comm_SensorDataReceived zpracovávající tyto data. Uloží se čas komunikace, ten se zobrazuje ve StatusBaru hlavního okna. Čas poslední komunikace se ukládá z toho důvody, aby v případě chyby bylo jasné, kdy byla naposledy přijata správná data. Nastaví se barva pro signalizační kolečko znamenající příjem paketu. Na základě přijatého Id se vytvoří objekt konkrétního senzoru a předají se data pro jejich zobrazovací komponenty (v našem případě StatusBary). Po zmáčknutí tlačítka vypnutí dojde k zastavení komunikace a odstranění barvy v signalizačním kolečku. Při samotném zavření hlavního okna dojde k ukončení vlákna obsluhující signalizační kolečko. V případě, že komunikace také stále běží dojde k jejímu ukončení.

4. Testování

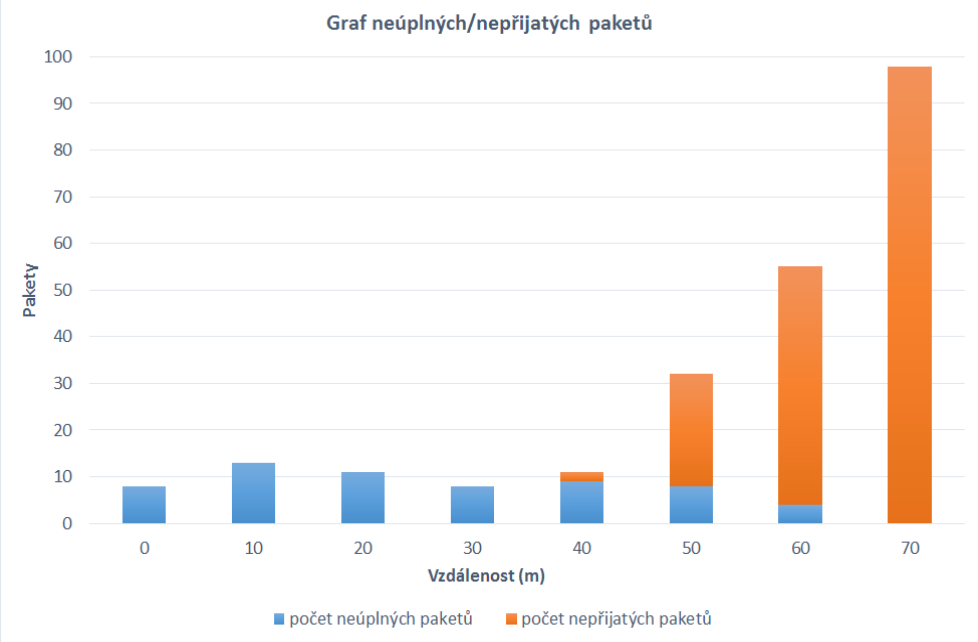
Jakmile jsme zdárně dokončili všechny předešlé úkoly, je potřeba otestovat v praxi funkčnost celého modulu pro snímání veličin, zpracovávání, posílání a zobrazování hodnot. Úkolem testování bylo změřit, na jakou vzdálenost je modul schopen odesílat a přijímat data, případně v jakém stavu se budou přijatá data nacházet. Princip testování byl následovný. První část modulu, v našem případě se jedná o senzor akcelerometru, vysílač a Arduino desku, byla umístěna na pevný bod a připojena k 9V baterii. Od této doby nebylo s touto části modulu nijak pohybováno. Tato první část snímá a odesílá data z akcelerometru. Druhá část modulu v našem případě jde od přijímač spolu s Arduino deskou byla připojena k notebooku, aby bylo možné sledovat v jakém stavu byla data přijata. Tato druhá část se umístila vždy do určité vzdálenosti od první a spustil se přenos dat podobu 100 s. Po tuto dobu se v intervalu 1 s odesílaly pakety. Po uplynutí 100 s se přenos zastavil a zpracovaly se naměřené hodnoty.

Jako první byl testován přenos dat na maximální vzdálenost. Mezi oběma částmi modulu byla přímá viditelnost, rychlost toku dat byla nastavena na 1 Mbps.



Obrázek 4.1: Graf správně přijatých paketů pro 1 Mbps

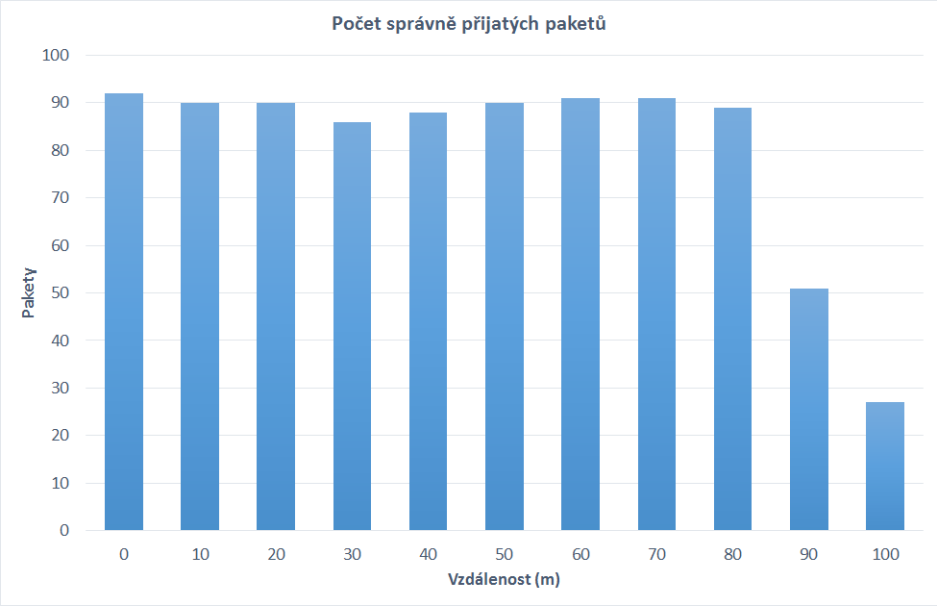
Mezi správně přijaté pakety jsem zařadil takové, které byly úplné, měly správné CRC a stop byty a nevykazovaly žádné prudké změny hodnot senzoru. Do vzdálenosti 40 m nebyly pozorovány výrazné rozdíly v přenosu. Od vzdálenosti 50–70 m začalo ubývat správných paketů a ve vzdálenosti 80 m nebyly schopny části modulů navázat komunikaci. Počty neúplných či nepřijatých paketů jsou vidět na následujícím obrázku.



Obrázek 4.2: Graf neúplných/nepřijatých paketů pro rychlost 1 Mbps

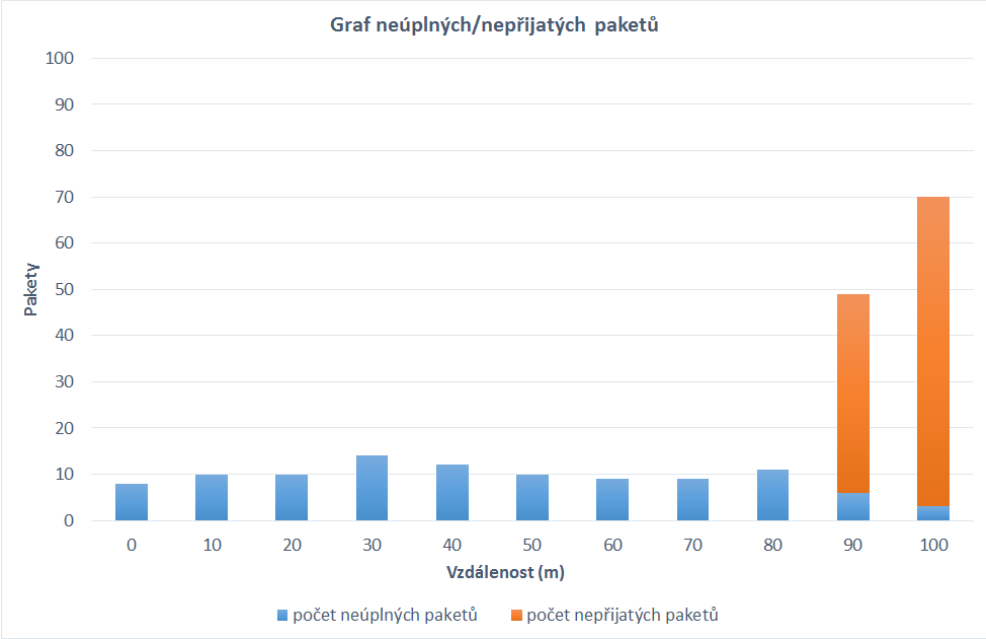
Z grafů vyplývá, že při rychlosti toku dat 1Mbps je schopen náš modul komunikovat do vzdálenosti 40–50 m, kde se počet neúplných či nepřijatých paketů pohybuje kolo 10 %. Od této vzdálenosti přibývá nepřijatých paketů. Maximální hodnota nepřijatých paketů dosahuje hodnoty 98 a to ve vzdálenosti 70 m.

Druhý test byl proveden na obdobném principu měření přenosu dat na maximální vzdálenost, tentokrát však s rychlostí toku dat 250 Kbps.



Obrázek 4.3: Graf správně přijatých paketů pro rychlost 250 Kbps

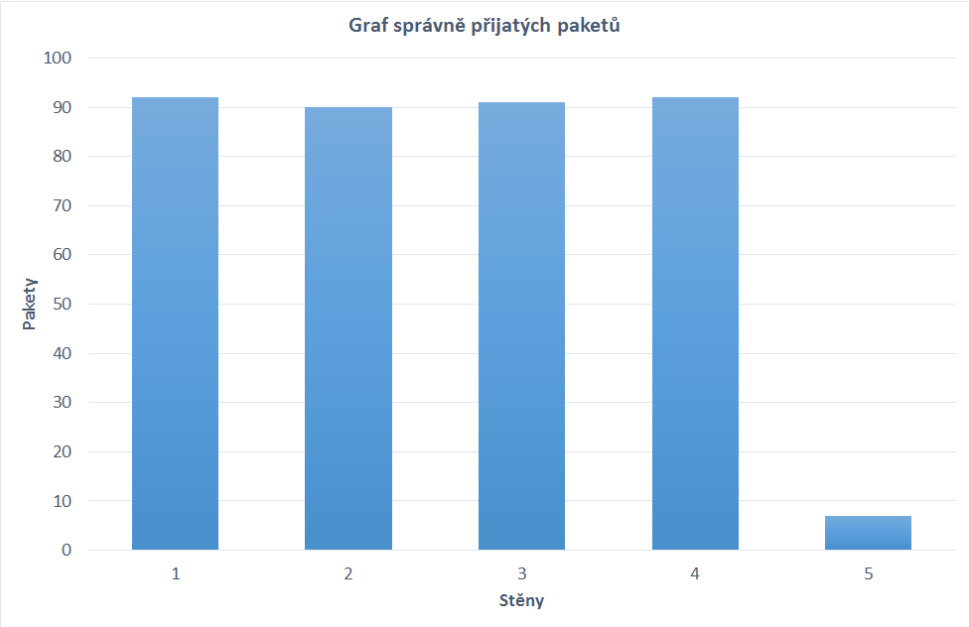
V grafu můžeme vidět, že změnou rychlosti toku dat došlo k prodloužení vzdálenosti pro komunikaci ze 40 na 80 m. Tento nárůst je také důsledkem změny citlivosti přijímače. Hodnota citlivosti pro rychlost 1 Mbps byla -85 dBm. Hodnota citlivosti se pro rychlost 250 Kbps zlepšila na -94 dBm. Určitý vliv na prodloužení vzdálenosti mělo také lepší umístěni obou části modulu do větší výšky od země, oproti měření s rychlosti toku dat 1 Mbps.



Obrázek 4.4: Graf neúplných/nepřijatých paketů pro rychlost 250 Kbps

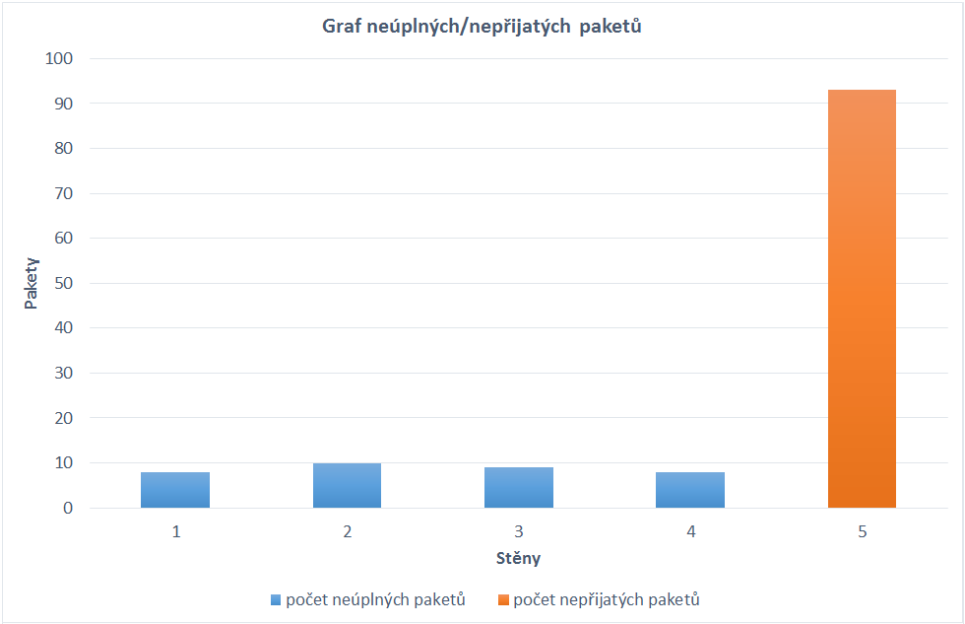
Z tohoto grafu je patrné, že změna rychlosti toku dat nemá vliv na počet neúplných paketů, i v tomto případě se pohybuje okolo 10 %. Počet nepřijatých paketů začat stoupat a to velmi výrazně až u vzdálenosti 90 m. Pro vzdálenost větší než 100 m se modulu nepodařilo navázat spojení.

Třetí a poslední test probíhal v rodinném domě. Testoval jsem schopnosti modulu posílat data skrz zdi. Princip byl opět podobný předchozím dvěma měřením. První část modulu byla umístěna na pevném bodě a druhá část modulu byla připojena k notebooku. Pohyboval jsem se s ní mezi jednotlivými místnostmi domu. Zdi jsou vyrobeny z cihlového materiálů. Tento materiál nemá takový vliv na útlum signálu jako železobetonové panely. Cihlové zdi jsou široké 20 cm až 25 cm. V každé místnosti byly odesílány pakety v intervalu 1 s po dobu 100 s. Rychlost toku dat zůstala na hodnotě 250 Kbps z toho důvodu, abych otestoval maximální možnou vzdálenost přenosu přes stěny. Tímto měřením jsem chtěl nasimulovat přenos dat ze senzorů, které se používají v domech jako například senzor teploty, vlhkosti atd. Tyto senzory většinou nejsou umístěny ve stejné místnosti jako zařízení pro zobrazení měřených dat, ale jsou venku či v jiné mistrnosti. Výsledné grafy můžeme vidět na následujících obrázcích.



Obrázek 4.5: Graf správně přijatých paketů skrz zdi pro rychlost 250 Kbps

Na tomto grafu je vidět, že modul dokáže bez problému komunikovat přes 4 cihlové stěny. Přes pátou stěnu prošlo pouze 7 paketů. Vzdálenost první části modulu od druhé byla v případě čtyř stěn 10 m a v případě pěti stěn 13 m.



Obrázek 4.6: Graf neúplných/nepřijatých paketů skrz zdi pro rychlost 250 Kbps

Z grafu je patrné, že ani stěny nemají vliv na počet neúplných paktů, který se i v tomto případě pohybuje na hranici 10 %.

Výsledky testovací části mě mile překvapily především pak schopnost modulu komunikovat na určitou vzdálenost. Pro rychlost 1 Mbps je tato vzdálenost okolo 40 m a pro rychlost 250 Kbps okolo 80 m. Výrobcem udávaná vzdálenost na kterou jsou schopny transceivery komunikovat je okolo 100 m. Tyto typy transceiveru existují také s anténou pro komunikaci na větší vzdálenosti. Vzdálenost se poté pohybuje pro rychlost toku dat 250 Kbps až okolo 1 km.

Obrázek 4.7: Transceiver Nrf24L01 s anténou pro přenos dat na delší vzdálenosti

Bohužel z grafů také vyplývá, že počet neúplných paketů, do kterých se řadí mimo jiné také pakety se špatným CRC a stop bytem se nemění s přibývající, či ubývající vzdálenosti přijímače od vysílače. Tato hodnota se pohybuje okolo 10 % všech přijatých paketů. Zde vidím největší prostor pro další zlepšení a práci na tomto modulu.

Závěr

Cílem mé bakalářské práce bylo navrhnout a realizovat hardwarový modul pro vzdálený přenos telemetrických dat. Toto zadání se mi podařilo splnit a vytvořit modul pracující na podobném principu jako dražší komerční systémy. Průběh tvorby práce probíhal až na drobné problémy dobře.

Začal jsem s rešerší telemetrických systému. Dozvěděl jsem se, jaké existují druhy a na jakém principu fungují. V jakých frekvenčních pásmech je možno s těmito systémy komunikovat a za jakých podmínek. Poté došlo k výběru jednotlivých hardwarových součástek. Jako první jsem řešil výběr RF modulů. Vybrané RF moduly mě velice mile překvapily. Nečekal jsem, že za takovou cenu lze získat takto schopná zařízení s nejrůznějším možným nastavením, které jsou navíc podporovány celou řadou knihoven a uživatelskými programy. Po výběru modulů přišel na řadu senzor pro snímání telemetrických veličin. Mým cílem bylo vybrat takový senzor, abych mohl snímána data nejen číst, ale také nějakým způsobem zpracovat. V mládí mě zaujaly závody Formule 1 a malých letadel, kde se zobrazovaly hodnoty přetížení, a už tehdy mne tato veličina zaujala. Proto při výběru senzoru padla volba na jeden z akcelerometrů spolu s gyroskopem. Tvorba bakalářské mi umožnila zjistit na jakém principu akcelerometr pracuje, a vyzkoušet si zpracovat jeho data způsobem, který se využívá při pilotování dronů či RC letadel. Avšak největší přínos pro mne měla práce s platformou Arduino. Naučil jsem se jak psát v jazyce pro tuto platformu, jak připojovat jednotlivá zařízení pomocí sběrnic a jakým způsobem s nimi komunikovat.

Po výběru všech zařízení a následném zapojení, přišla na řadu softwarová část. Při výběru jednotlivých knihoven pro senzor a RF moduly jsem dal přednost knihovně vycházející z technologie ZigBee. Tím vznikla komplikace, která byla odhalena ke konci bakalářské práce. Knihovna používaná pro obsluhu modulů bohužel neumožňovala otestovat a použít veškeré možnosti nastavení, jimiž moduly disponovaly. Po výběrů knihoven jsem začal s návrhem uživatelské části paketu. Zde vznikla další komplikace. První podoba uživatelské části paketu nebyla vyhovující, program odesílající uživatelskou část byl natolik velký a neúsporný, že jeho nahrání do jedné z Arduino desek nebylo možné. Program nenabízel možnost dalšího přidání senzorů a jakékoliv přizpůsobení uživateli. Proto jsem musel celou uživatelskou část paketu přepracovat, zobecnit a zpřehlednit do finální podoby. Nyní už zbývalo vytvořit software pro zobrazování měřených hodnot. Tento software byl tvořen tak, aby bylo uživateli naprosto zřejmé jaký senzor je aktuálně používán a jaké měří hodnoty. Závěrečné testování ukázalo, že navržený modul je schopen komunikovat do vzdáleností přibližující se testovaným vzdálenostem výrobce RF modulu.

V další fázi nabízí bakalářská práce možnost změny knihoven pro RF moduly. Dále nabízí prostor pro zlepšování v oblasti spolehlivosti celého modulu a jeho přizpůsobení pro další typy senzorů.

Použitá literatura

[1] *NRF24L01 Single Chip 2.4GHz Transceiver Product Specification* [online]. 2007 [cit. 2015-04-28]. Dostupné z: http://www.nordicsemi.com/eng/Products/2.4GHz-RF/nRF24L01

[2] JURÁNEK, Martin. Prostředky automatického řízení: Akcelerometry: *Homel.vsb* [online]. 2007 [cit. 2015-04-28]. Dostupné z: http://homel.vsb.cz/~jur286/prostredky_aut_rizeni/preklad.htm

[3] INVENSENSE. *MPU-6000 and MPU-6050 Product Specification* [online]. ¨2013 [cit. 2015-04-28]. Dostupné z: http://www.invensense.com/mems/gyro/documents/PS-MPU-6000A-00v3.4.pdf

[4] *Arduino* [online]. 2015 [cit. 2015-04-28]. Dostupné z: http://www.arduino.cc/

[5] LANGBRIDGE, James A. *Arduino sketches: tools and techniques for programming wizardry* [online]. 2015, pages cm [cit. 2015-04-28]. ISBN 11-189-1960-2. Dostupné z: www.wiley.com

[6] ATMEL. *ATmega328p datasheet summary* [online]. 2009 [cit. 2015-05-01]. Dostupné z: http://www.atmel.com/Images/Atmel-8271-8-bit-AVR-Microcontroller-ATmega48A-48PA-88A-88PA-168A-168PA-328-328P_datasheet_Summary.pdf

[7] ATMEL. *ATmega2560 datasheet summary* [online]. 2014 [cit. 2015-05-01]. Dostupné z: http://www.atmel.com/Images/Atmel-2549-8-bit-AVR-Microcontroller-ATmega640-1280-1281-2560-2561_Summary.pdf

[8] MOTOROLA. *SPI Block Guide* [online]. 2000 [cit. 2015-04-28]. Dostupné z: http://www.ee.nmt.edu/~teare/ee308l/datasheets/S12SPIV3.pdf

[9] NXP SEMICONDUCTORS. *I2C-bus specification and user manual* [online]. 2014 [cit. 2015-04-28]. Dostupné z: http://www.nxp.com/documents/user_manual/UM10204.pdf

[10] Arduino Development Environment. *Arduino* [online]. 2014 [cit. 2015-04-28]. Dostupné z: http://www.arduino.cc/en/Guide/Environment

[11] [ROWBERG, Jeff. MPU-6050. Github [online]. 2011 [cit. 2015-04-01]. Dostupné z: https://github.com/jrowberg/i2cdevlib/tree/master/Arduino/MPU6050

[12] COLIZ, James. Network Layer for RF24 Radios [online]. 2011 [cit. 2015-04-04]. Dostupné z:http://maniacbug.github.io/RF24Network/index.html

[13] PETERSON, W. a D. BROWN. *Cyclic Codes for Error Detection*. New York: United Nations, 1987, ix, 499 s. ISSN 0096-8390.